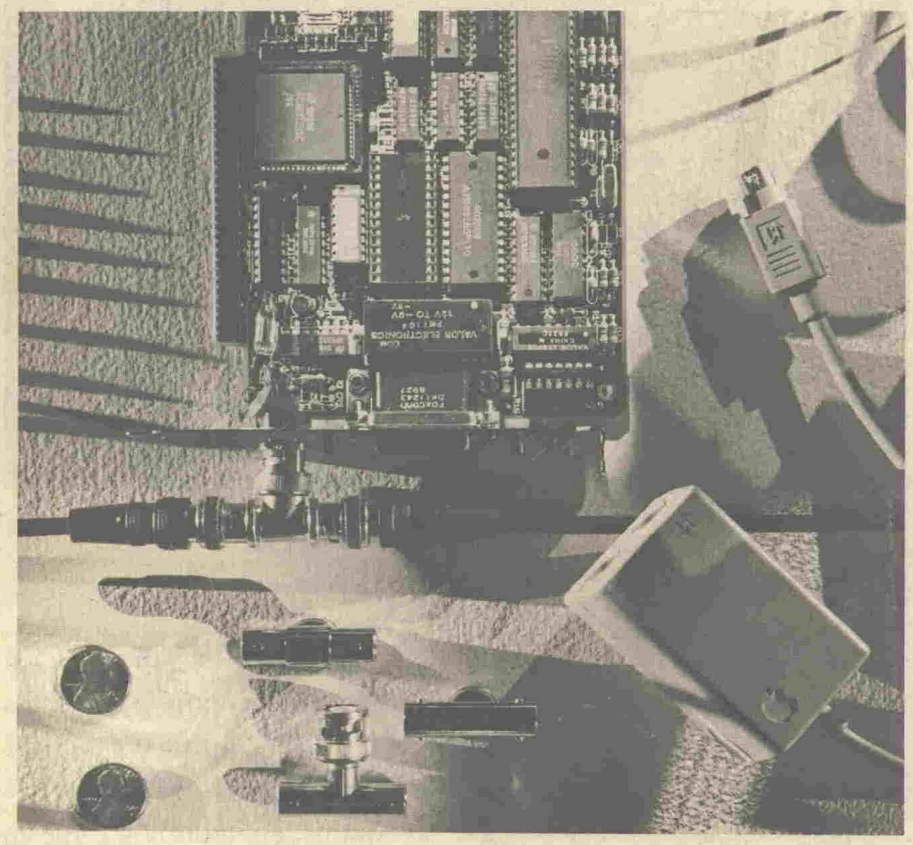
เรื่องน่าอ่าน
ดร.ปัญญา เปรมปรีดิ์

# สัมผัสที่ 6 กับ ระบบแลน (จบ)



***ระบบกึ่งศูนย์รวม และกึ่งกระจายศูนย์ รวมกันในไฟล์เซอร์ฟเวอร์***

ท่านผู้อ่านคงได้รับความกระจ่างและสัมผัสเพียงบางส่วนของระบบแลนจากตอนที่ 1 ไปบ้างแล้ว จากเคยที่คิดว่ามีเครื่องไมโครคอมพิวเตอร์อยู่หลายตัวแล้วสามารถเชื่อมโยงกันเป็นแลนได้นั้น และการเอาแลนมาใช้เพื่อรับส่งไฟล์ระหว่างผู้ใช้เครื่องไมโครคอมพิวเตอร์นั้นไม่คุ้มค่า สำหรับตอนจบจะขอเสนอต่อและเจาะลึกถึงเทคนิคการออกแบบระบบ และระบบยูนิกซ์ซึ่งเป็นระบบที่ใกล้เคียงกับแลนมากที่สุด

การใช้ลอจิกแบบที่กล่าวมาตอนที่นั้นเป็นวิธีการแก้ปัญหาที่สั้นและง่แต่ก็อาจมีข้อจำกัดอยู่บางประการ กล่คือเมื่อเรามีปริมาณงานเพิ่มมากขึ้น เอาจต้องรอคอยการเปิดไฟล์กันอีก เรื่นี้ขึ้นอยู่กับระยะเวลาที่ต้องใช้ในขั้นตอที่ (2) ถ้าเราบีบให้ระยะเวลาในขั้นตอนสั้นลงได้ เราก็จะทำงานขนานกันได้มาเทอร์มินัล แต่ถ้าระยะเวลาในขั้นตอนี้ยาวเราก็จะทำงานขนานกันได้น้อยตัวเลขคร่าว ๆ จะเป็นดังนี้

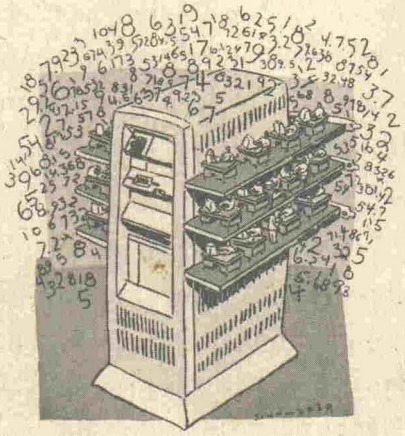
สมมติว่าการอัพเดตในขั้นตอที่ (2) นี้ต้องเปิดไฟล์ 3 ไฟล์ซึ่งต้องอ่และบันทึกไฟล์ละ 4 ครั้ง เราก็ต้องเวลาอ่านและบันทึก 3×4×20 = 2มิลลิเซกชัน นี่เพียงสมมติว่าการอ่หรือบันทึกแต่ละครั้งใช้เวลา 20 มิลลิเชชัน นอกจากเวลาในการอ่านและบันทีจานแม่เหล็กแล้วเรายังมีการคำนวอยู่ในขั้นตอนที่ (2) นี้ด้วย เรื่องนี้สมมว่าใช้เวลา 0.2 วินาที ตัวเลขนี้มาจาการตั้งสมมติฐานว่ามีการเอกซีคิ(Execute) คำสั่ง 200,000 คำสั่ง แเครื่องไมโครคอมพิวเตอร์ที่เป็นเทอร์มิของเรามีความเร็ว 1 ล้านคำสั่งต่อ 1 วิน(1 MIPS) จากตัวเลขทั้ง 2 จะประมาได้ว่าขั้นตอนที่ (2) นี้ใช้เวลา 0.44 วินจากตัวเลขนี้เทียบกับเวลาในขั้นตอนที่ (บวกกับ (3) จะเป็นอัตราส่วนประมา1 ต่อ 100 ตัวเลขอัตราส่วน 1 ต่อ 100มีความหมายคือ ถ้ามีคนใช้ 1 เทอร์มิอย่างต่อเนื่องก็จะมีการล็อกไฟล์อยู่อัตราประมาณ 1/100 ของ เวลาทั้งหมและถ้ามีคนใช้อยู่ n เทอร์มินัลก็จะการล็อกไฟล์อยู่ในอัตราประมาณ n/1ของเวลาทั้งหมด ดังนั้นถ้า n มีค่ามากก100 เราก็จะเห็นว่าต้องมีการรอคอยแน่ ๆ เรื่องนี้จริง ๆ แล้วจะเริ่มเห็ผลกระทบเมื่อ n มีค่าประมาณ 60 ขึ้น

ตัวเลขข้างบนนี้เป็นตัวเลขของ
ะบบงานที่ออกแบบได้ระดับดีพอสมควร
านทั่ว ๆ ไปจะต้องอ่านและบันทึก
านแม่เหล็กไม่เกิน 12 ครั้งต่อ 1 รายการ
้าออกแบบได้ดีอย่างนี้ก็จะใช้กับแลน
นาด 60-70 เทอร์มินัลได้ แต่ในกรณีที่
ปรแกรมเมอร์ของเราไม่มีประสบการณ์
าอาจออกแบบลอจิกของโปรแกรม
ดยไม่ได้คำนึงถึงเรื่องนี้และใช้การอ่าน
ละบันทึกถึง 50-60 ครั้ง ยิ่งเป็นพวก
ใช้ภาษาฐานข้อมูลจะยิ่งมีการอ่านและ
ันทึกสูงมาก 1 รายการอาจเกิดการอ่าน
ละบันทึกถึง 200 ครั้ง ผลจะออกมา
่ดีอย่างไม่น่าเชื่อ คือเวลาในขั้นตอน
(2) จะเพิ่มเป็น 200×20+200 = 4,200
ลลิเซกชัน อัตราส่วนของเวลาที่ไฟล์
กล็อกต่อไฟล์ไม่ถูกล็อกจะเปลี่ยนไป
ป็นประมาณ 1 ต่อ 10 ระบบก็จะใช้
ทอร์มินัลได้เพียง 6-7 เทอร์มินัลเท่านั้น
งเป็นปัญหาที่เราเห็นกันได้บ่อยที่สุด
ื่อใช้ภาษาดีเบสในระบบแลนงานแบบนี้
ะใส่เทอร์มินัลได้แค่ 6-7 ตัว ใส่มากกว่า
จะทำงานช้าลงแล้วก็มานั่งบ่นนั่งต่อว่า
ะบบแลน เรื่องนี้ผู้เขียนขอแนะนำให้
ช้ซอฟต์แวร์ที่มีประสิทธิภาพมากกว่านี้
ัน BTREIVE จะต้องบังคับโปรแกรม-
อร์ให้รักษากฎเกณฑ์ในการเขียนโปร-
กรมที่รัดกุม การอ่านและบันทึกจาน-
่แม่เหล็กต้องไม่เกิน 12 ครั้งต่อ 1 รายการ
งแลนที่ผู้เขียนใช้อยู่นั้นบริการงาน
นาด 25 เทอร์มินัลได้สบาย ๆ

นั่นเป็นข้อจำกัดของเทคนิคแบบ
เปิดไฟล์แล้วล็อกเอาไว้ใช้แต่เพียงผู้เดียว
ป็นระยะเวลาหนึ่ง" แต่จากตัวเลขที่
เคราะห์ให้ดูก็จะเห็นว่ายังใช้กับแลน
นาด 60-70 เทอร์มินัลได้ ซึ่งจะใหญ่
นพอสำหรับงานในเมืองไทยเรา แต่
ธีการข้างบนนี้ก็ยังไม่ใช่วิธีที่ดีที่สุด

วิธีที่ดีกว่านี้คือซื้อแลนที่เปิดโอกาสให้เราแชร์ไฟล์กันได้จริง ๆ แล้วไปล็อกเรคอร์ดให้เรา การล็อกเรคอร์ดนี้จะกระทำโดยอัตโนมัติ คือในคำสั่ง READ นั้นจะไม่ต้องบอกว่า "LOCK" โอเปอเรติ้งซิสเท็มจะจัดการล็อกให้เราเลยและจะปล่อยเรคอร์ดนี้เมื่อใช้คำสั่ง WRITE หรือ RELEASE

ข้อดีของระบบล็อกในระดับเรคอร์ดคือ ทุกโปรแกรมหรือทุกเทอร์มินัลจะทำงานขนานกันได้เมื่อกำลังอัพเดตเรคอร์ดที่ไม่ตรงกัน ด้วยวิธีนี้จะทำให้การรอคอยระหว่างเทอร์มินัลจะลดน้อยลงมาก การรอคอยกันจะเกิดขึ้นเมื่อจะต้องใช้เรคอร์ดเดียวกันพอดีเท่านั้น ซึ่งปกติแล้วจะมีโอกาสน้อยมาก แต่บางทีโปรแกรมเมอร์มือใหม่ ๆ ก็สร้างปัญหาให้เราได้การออกแบบผิด ๆ ที่พบบ่อยที่สุดคือ เรามักจะให้มีไฟล์หรือเรคอร์ดอยู่อันหนึ่ง เพื่อเก็บตัวเลขบางชนิด เช่น "หมายเลขสลิปอันล่าสุดที่เอาไปใช้แล้ว" ตัวเลขชนิดนี้ต้องใช้ทุกครั้งที่มีการอัพเดตโปรแกรมทุกโปรแกรมจึงต้องมาอ่านและอัพเดตเรคอร์ดนี้ เทคนิคการออกแบบระบบแบบนี้ใช้กันในงานทั่วไป หมายเลขสลิปจะถูกสร้างโดยโปรแกรมและถูกใช้กับสลิปโดยเรียงลำดับไป หมายเลขสลิปอาจถูกรีเซตเป็น 00001 ในทุก ๆ เช้าของวันทำงาน หมายเลขสลิปจะถูกพิมพ์ลงในสลิปและถูกเก็บไว้ในแฟ้มข้อมูล ทั้งนี้เพื่อเป็นตัวเลขอ้างอิง บางทีก็ต้องใช้เป็นคีย์ในการเรียกหาข้อมูลเมื่อมาติดต่อในวันหลัง นอกจากหมายเลขสลิปแล้วก็ยังมีตัวเลขแบบเดียวกันอีกมากมาย เช่น หมายเลขประจำรายการ (Transaction Number) หมายเลขประจำเช็ค (Cheque Running Number) หมายเลขประจำตัวลูกค้า (Account Number) ตัวเลขเหล่านี้จะถูกอ่านขึ้นมาแล้วเอา 1 บวกเข้าไปตัวเลขใหม่ก็จะถูกใช้ติดไปกับข้อมูลการอัพเดต แล้วเรคอร์ดที่เก็บตัวเลขนี้ก็จะถูกบันทึกกลับลงไปที่จานแม่เหล็ก



ถ้าเรามีเรคอร์ดแบบตัวร่วมขึ้นมา ปัญหาเก่าก็จะเกิดขึ้นคือ แทนที่จะต้องรอการเปิดไฟล์ เราก็ต้องมาเข้าคิวรอใช้เรคอร์ดที่เป็นตัวร่วมเหล่านี้ ปัญหาจะไม่แตกต่างกันเลยแต่คราวนี้ต้องตามไปแก้ไขโปรแกรมหลายที่ซึ่งเรามักจะจับกันไม่ค่อยหมด จะมีโปรแกรมบางโปรแกรมหลงหูหลงตาอยู่ ซึ่งจะทำให้เครื่องมีอาการ "ทำงานช้าเป็นบางครั้งบางคราว" และหาสาเหตุไม่ออก บางทีเราไปเรียกช่างผู้เชี่ยวชาญมาช่วยกันนั่งจับเป็นสิบ ๆ วันก็ยังมองไม่เห็นที่เป็นเช่นนี้ก็เพราะมันไม่เกิดขึ้นบ่อยนัก และเราไม่เคยคิดมาก่อนว่าจะมีโปรแกรมเมอร์คนไหนหลงลืมเราไปคิดว่าทุกคนเข้าใจแล้ว แต่ในทางปฏิบัติก็จะพบว่าโปรแกรมเมอร์นี่ทำผิดได้เสมอเขาก็เป็นมนุษย์และปุถุชนเช่นกัน

นอกจากจะเขียนโปรแกรมโดยอ่านเรคอร์ดที่เป็นตัวร่วมแล้วไม่ปล่อยทันทีแล้ว เรายังจะพบอีกว่าโปรแกรมเมอร์เหล่านี้จัดลำดับการอ่านเรคอร์ด

ไม่ตรงกันอีกด้วย ซึ่งจะทำให้เกิดการ "ล็อกค้าง (Interlocked)" ขึ้นมาได้ ยกตัวอย่างเช่น โปรแกรมที่ทำรายการชนิด A อาจอ่านหมายเลขสลิปก่อนแล้วจึงมาอ่านหมายเลขรายการแต่โปรแกรมที่ทำรายการ B ซึ่งเขียนโดยโปรแกรมเมอร์อีกคนหนึ่งอาจสั่งให้อ่านหมายเลขรายการเสียก่อน แล้วจึงไปอ่านหมายเลขสลิปทีหลัง สองโปรแกรมนี้อาจใช้พร้อม ๆ กันไม่บ่อยนักและถึงแม้ว่าจะใช้พร้อมกัน แต่ก็อาจใช้เวลาในขั้นตอนเหล่านี้เร็วมาก จึงทำให้เวลาไม่คาบเกี่ยวกันพอดี แต่ก็ไม่ช้าไม่นาน (5 วัน 10 วัน 100 วัน) ก็จะเกิดเหตุการณ์พิเศษ คือโปรแกรม A อ่านเรคอร์ดที่เก็บหมายเลขสลิปขึ้นไปแล้วและกำลังจะอ่านเรคอร์ดที่เก็บหมายเลขรายการ ขณะเดียวกันโปรแกรม B ก็อ่านเรคอร์ดที่เก็บหมายเลขรายการขึ้นไปแล้วและกำลังจะอ่านเรคอร์ดที่เก็บหมายเลขสลิป ในสถานการณ์เช่นนี้ทั้ง 2 โปรแกรมจะต้องหยุดรอและจะรอกันไปโดยไม่มีที่สิ้นสุด นี่คืออาการ "ล็อกค้าง" ซึ่งจะส่งผลทำให้ทุกโปรแกรมและทุกเทอร์มินัลมาหยุดรอกันหมด เพราะพวกนี้ในไม่ช้าก็ต้องวิ่งไปถึงจุดที่จะขออ่านเรคอร์ดที่เก็บหมายเลขสลิปหรือหมายเลขรายการขึ้นมาบ้าง แต่เรคอร์ดทั้ง 2 ยังถูกล็อกค้างไว้โดยโปรแกรม A และ B เรื่องแบบนี้เกิดขึ้นไม่บ่อยนักแต่ก็เกิดขึ้นได้ การออกแบบลอจิกของโปรแกรมจึงต้องบังคับว่า "ให้จัดลำดับการอ่านเรคอร์ดชนิดต่าง ๆ ให้ตรงกันทุกโปรแกรม" ด้วยวิธีนี้จะไม่ทำให้เกิดการ "ล็อกค้าง"

นี่คือหลักการออกแบบระบบ ถ้าเราป้องกันการล็อกค้างไปได้แล้วก็จะเกิดการรอคอยกันเพียง 20-40 มิลลิเซกชัน อัตราส่วนระหว่างเวลาที่เรคอร์ดถูกล็อกกับไม่ถูกล็อกจะสูงขึ้นมาก เช่น 1:500 ถึง 1:1000 จำนวนเทอร์มินัลที่จะทำงานขนานกันได้ก็จะขยายไปเป็น 400-600 เทอร์มินัล การใช้แลนจึงไม่มีขีดจำกัดในด้านนี้

ทั้งหมดที่เล่ามานี้ท่านผู้อ่านอย่าคิดว่าเป็นปัญหาของแลนอย่างเดียว เพราะการออกแบบระบบออนไลน์-เรียลไทม์ในเครื่องเมนเฟรมและมินิคอมพิวเตอร์เป็นอย่างเดียวกัน พวกนั้นยิ่งซ้ำร้ายกว่าคือเวลาที่ใช้ในการคำนวณในขั้นตอนที่ (2) ที่กล่าวมาแล้วนั้นจะต้องรวมถึงการรอคอยกันใช้ซีพียูด้วย ระบบเมนเฟรมและมินินั้นต้องคำนวณด้วยซีพียูตัวเดียวกันหมด การป้องกันการรอคอยซึ่งกันและกันจึงมีมากกว่า ระบบเมนเฟรมและมินิคอมพิวเตอร์นั้นต้องทำงานเป็นชนิด "ทีละรายการ" และต้องให้เราส่งชื่อโปรแกรมเข้าไปใหม่ทุกครั้ง เพราะเมื่อจบในขั้นตอนที่ (2) แล้วนั้นจะต้องให้จบโปรแกรม (Terminate) ด้วย ทั้งนี้เพื่อลดการใช้เนื้อที่ในเมมโมรี แต่ระบบแลนนั้นไม่ต้องจบโปรแกรมเรายังให้โปรแกรมทำงานวนต่อไปได้เรื่อย ๆ การโต้ตอบกับจอภาพจะทำกี่ครั้งก็ได้ ทั้งนี้เพราะตัวโปรแกรมจะมาอยู่ที่เทอร์มินัลการรับข้อมูลจึงทำได้ง่ายกว่าและดีกว่าเมนเฟรมและมินิคอมพิวเตอร์ เช่น การอีดิติ้งข้อมูลนั้นจะทำอยู่ที่หน้าจอภาพนานเท่าไหร่ก็ได้ มันจะไม่กระทบกับโปรแกรมในเทอร์มินัลอื่นและไม่มีการแย่งกันใช้เมมโมรี

มาถึงตรงนี้ก็อยากจะชี้สรุปอะไรบางอย่างเอาไว้ คือระบบแลนนี้เป็นระบบกึ่งรวมศูนย์ (Centralized) และกึ่งกระจายศูนย์ (Distributed) ซึ่งมีลักษณะรวมศูนย์ก็เพราะแฟ้มข้อมูลที่ต้องใช้ร่วมกันนั้นจะรวมอยู่ที่ไฟล์เซอร์เวอร์ แต่ขณะเดียวกันโปรแกรมที่จทำงานนั้นกลับกระจายอยู่ในเครื่องไมโที่เป็นเทอร์มินัล การคำนวณจึงกระจากันอยู่ ตัวที่ต้องระวังจึงลดลงจากระบรวมศูนย์ ระบบเมนเฟรมและมินิคอมพิวเตอร์ที่มีหัวสมองเดี่ยว ๆ นั่นเป็ระบบรวมศูนย์ ซึ่งต้องระวังไม่ให้โปรแกรมใดไปครอบครองข้อมูลและเมมโมแต่เพียงผู้เดียวทุกคนต้องหมุนวนกัเข้าไปใช้ทรัพยากร (CPU, DISK, I/Channel, etc.) โปรแกรมต้องเขียนอย่มีประสิทธิภาพมากและฮาร์ดแวร์ก็ต้อมีความเร็วสูง ราคาจึงแพง คิดค่าลงทุจะตกประมาณ 1 ล้านบาทต่อ 1 เทอร์มินระบบมินิและซูเปอร์มินิขนาดเล็กถูกลงมาหน่อยหนึ่ง คือเหลือประมา5 แสนบาทต่อ 1 เทอร์มินัล แต่กาทำงานจะโต้ตอบด้วยความเร็วประมา3 ถึง 5 วินาที สำหรับระบบแลนนั้จะลงทุนประมาณ 1.7 แสนบาทต่อเทอร์มินัล แล้วยังทำงานได้เร็วขึ้น คส่วนใหญ่จะกดแล้วได้ผลกลับมาในเวไม่ถึง 1 วินาที ใครที่พัฒนางานออนไล-เรียลไทม์ลงระบบแลนแล้วโต้ตอบช้ากว่า 1 วินาทีก็จะต้องไปเช็คดู เพรามักจะเกิดจากการออกแบบผิดหรือกี่เขียนโปรแกรมไม่ถูกหลัก

อยากจะขอย้ำอีกครั้งหนึ่งว่าระแลนไม่ใช่ระบบรวมศูนย์ (Centralize มันแชร์กันใช้ไฟล์ที่อยู่ในตัวไฟล์เซอร์เวอร์ก็จริง แต่มันก็ไม่ใช่ระบบรวมศูและยิ่งไปกว่านั้น ในระบบแลนนี้สามารถใช้จานแม่เหล็กและอุปกรณ์อื่นที่เทอร์มินัลได้หมดและการทำงานอุปกรณ์เหล่านี้ไม่มีผลกระทบต่อกทำงานของเทอร์มินัลอื่น เราจึงมีควคล่องตัวในการออกแบบและเขียนโป

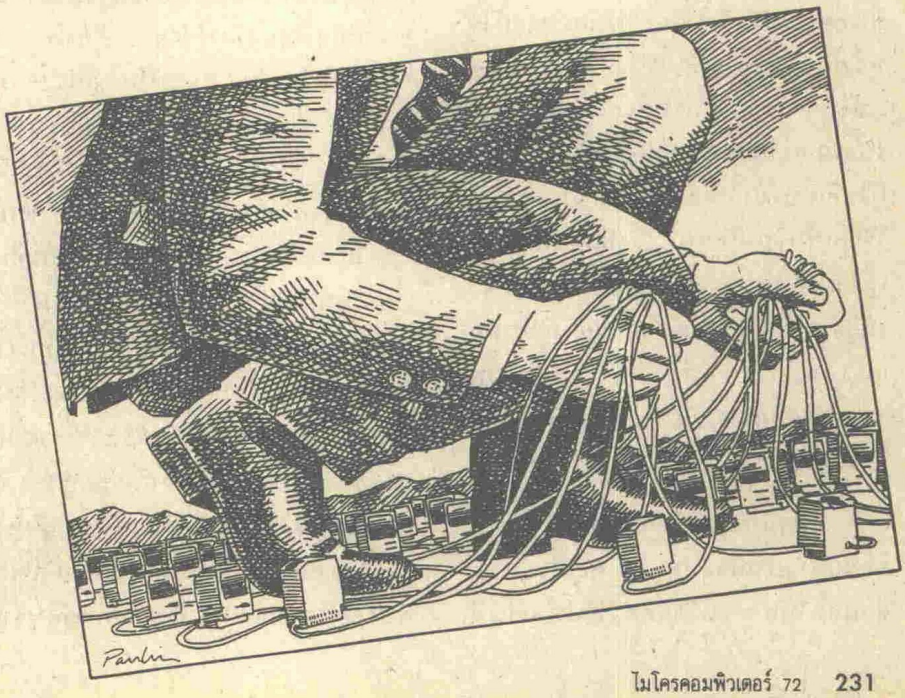
กรมอีกมาก การเอาเครื่องไมโคร-
อมพิวเตอร์ไปต่อเป็นเทอร์มินัลของ
นเฟรมและมินิคอมพิวเตอร์นั้นก็ไม่
หมือนแลน เพราะโปรแกรมที่อีมูเลต
Emulate) ให้เป็นเทอร์มินัลนั้นไม่สามารถ
ียกใช้จานแม่เหล็กและอุปกรณ์อื่น ๆ
เทอร์มินัลได้

## ะบบยูนิกซ์

ทีนี้เรามาพูดถึงระบบยูนิกซ์กันบ้าง
นิกซ์ที่ใกล้เคียงกับแลนมากที่สุดก็คือ
ช้ซูเปอร์ไมโครหรือไฮเพาเวอร์เวอร์ก-
เตชันเป็นตัวศูนย์กลาง 1 ตัว แล้วโยง
ายสื่อสารแบบความเร็วต่ำ 1200-9600
ตต่อ 1 วินาที ไปยังเครื่องไมโคร-
อมพิวเตอร์หรือดัมพ์เทอร์มินัลจำนวน
นึ่ง ตัวศูนย์กลางนี้จะทำหน้าที่ทั้งเป็น
ฟล์เซอร์ฟเวอร์และตัววิ่งโปรแกรมให้
บเทอร์มินัล ถ้าเราใช้เครื่องไมโคร-
อมพิวเตอร์ระดับ 80386 เป็นตัวศูนย์-
ลาง เราจะใส่เทอร์มินัลได้ประมาณ
ถึง 10 ตัวเท่านั้นมากกว่านี้จะทำงาน
มาก ๆ จนไม่น่าใช้ เราลองมาสมมติว่า
ช้เครื่องไมโครคอมพิวเตอร์ 10 ตัวเป็น
อร์มินัลดู เงินลงทุนจะเป็นประมาณ
9-2.1 ล้านบาท การที่หยิบยกเอา
ะบบยูนิกซ์ชนิดนี้ซึ่งเรียกว่า มัลติยูส-
อร์ ขึ้นมาเปรียบเทียบก็เพราะเรากำลัง
องถึงงานประเภทออนไลน์-เรียลไทม์
งเคยวิ่งอยู่ในเครื่องมินิคอมพิวเตอร์
รือเมนเฟรม ระบบยูนิกซ์จะแพงกว่า
ลนในเรื่องค่าซิสเต็มซอฟต์แวร์และ
าเมมโมรีที่ต้องใช้ในตัวศูนย์กลาง แต่
ะได้เปรียบในเรื่องการสื่อสาร เพราะ
ช้สายโทรศัพท์ภายในธรรมดา ๆ และ
งเทอร์มินัลได้ไกล ๆ

การประมาณเวลาในการทำงาน
ายในระบบยูนิกซ์นั้นจะเหมือนกับใน
ระบบมินิคอมพิวเตอร์ทุกประการ การออกแบบลอจิกของโปรแกรมจะมี 2 ประเภท ประเภทแรกคือ ให้โปรแกรมที่เราสั่งให้วิ่งนั้นวิ่งต่อเนื่องอยู่ตลอดเวลา คือเริ่มโหลดตอนเช้าใช้งานไปเรื่อย ๆ ทำงานเสร็จไปแต่ละรายการก็จะวนกลับไปตั้งต้นที่จุดเริ่มของรายการใหม่ เขาจะใช้ไปจนกว่าจะสั่งให้จบโปรแกรม ซึ่งมักจะทำเมื่อตอนเลิกงานการออกแบบลอจิกแบบนี้จะทำให้ไม่ต้องโหลดโปรแกรมบ่อย ๆ แต่โปรแกรมนี้ต้องทำรายการได้ทุกประเภท โปรแกรมจึงโตมาก เมื่อมีคนมานั่งใช้ครบทุกเทอร์มินัลก็จะเกิดการแย่งกันใช้เมมโมรี ถ้าเมมโมรีไม่พอที่จะโหลดได้ครบทุกเทอร์มินัลก็จะเกิดการโยกย้ายโปรแกรมลงไปเก็บในจานแม่เหล็ก ซึ่งเราเรียกกันว่าเวอชวลสตอเรจ (Virtual Storage) ซึ่งจะไปเพิ่มภาระให้กับระบบจานแม่เหล็กเพราะต้องมีการอ่านและบันทึกในจานแม่เหล็ก

ลอจิกของโปรแกรมในมินิคอมพิวเตอร์อีกแบบหนึ่งคือ ต้องทำงานให้เสร็จทีละรายการ หรือเป็น "Transaction Oriented" ลอจิกจะมีขั้นตอนดังนี้ คือหลังจากที่เราเชื่อมโยงกับระบบล็อกออนได้แล้วเราก็คีย์ชื่อของรายการและข้อมูลที่ต้องใช้ในการอัพเดตตามรายการชนิดนั้น แล้วกดเอ็นเตอร์เข้าไป ซิสเต็มซอฟต์แวร์จะไปโหลดโปรแกรมที่ตรงกับชื่อของรายการเข้ามา และส่งมอบข้อมูลที่ปะท้ายเข้ามานั้นให้ โปรแกรมนั้นก็จะคำนวณ อ่านไฟล์ อัพเดตไฟล์ แล้วส่งผลเพื่อขึ้นจอภาพ แล้วจบโปรแกรม (Terminate) ไปเลย วิธีการนี้มีข้อดีคือโปรแกรมที่เราเรียกใช้แต่ละครั้งนั้นเป็นโปรแกรมเล็ก ๆ ทำงานตามชนิดของรายการ (Transaction Type) ที่ต้องการเท่านั้น โปรแกรมจะถูกโหลดเข้ามาแล้วอยู่ในเมมโมรีเพียงไม่กี่วินาทีแล้วก็จบไป เมมโมรีจึงถูกใช้ในระยะเวลาสั้น ๆ เมื่อถูกปลดปล่อยแล้วก็ให้เทอร์มินัลอื่นได้ใช้บ้าง เมมโมรีรวมจึงใช้ไม่มากเครื่องมินิคอมพิวเตอร์รุ่นแรก ๆ มีเมมโมรีเพียง 1-2 เมกะไบต์ พวกเขามักต้องใช้แนวความคิดแบบนี้ แต่ลอจิกแบบนี้ทำให้การอินพุตข้อมูลนั้นไม่สวย คือ



ต้องคีย์เป็นตัวเลขต่อเนื่องกันเข้าไป คนรุ่นใหม่เกือบจะไม่รู้จักกันแล้วและมักจะเขียนตามลอจิกแบบแรกซึ่งจริง ๆ แล้วเราเรียกว่าเป็น "ระบบทำงานต่อเนื่อง" หรือ "Continuous Process" แต่พวกเขากลับไปคิดว่าเป็น "Transaction Oriented" เพราะออกแบบให้วนทำทีละรายการ คำศัพท์คำนี้ยังมีคนเข้าใจผิดกันอยู่มาก

เพื่อเป็นตัวอย่างจะขอสมมติว่าเราใช้เทคนิคที่คนรุ่นใหม่เขาชำนาญกันคือ "ทำงานทีละรายการ...แต่วนต่อเนื่องไปเรื่อย ๆ ทั้งวัน" โปรแกรมแบบนี้ต้องทำได้ทุกชนิดของรายการและจะมีการโต้ตอบกับผู้ใช้ที่เทอร์มินัลอยู่ 2-3 รอบก่อนเสร็จรายการ เช่น หลังจากโหลดโปรแกรมแล้ว โปรแกรมก็จะส่งภาพเมนเมนูไปขึ้นที่จอภาพ ผู้ใช้ที่เทอร์มินัลต้องเลือกชนิดของรายการ ซึ่งจะเกิดการส่งตัวอักษรเข้ามา 2-3 ตัว โปรแกรมตีความแล้วก็จะส่งภาพเมนูที่ตรงกับชนิดของรายการออกไป ผู้ใช้ที่เทอร์มินัลก็จะคีย์ข้อมูลต่าง ๆ เข้ามา เครื่องรับไปแล้วก็ทำการอีดิติ้งแล้วส่งผลกลับไปขึ้นที่จอภาพ ผู้ใช้ที่เทอร์มินัลต้องแก้ไขหรือกดบอกตกลง (Confirm) เข้ามา หลังจากนั้นโปรแกรมก็จะอัพเดตข้อมูลในจานแม่เหล็กแล้วส่งผลไปขึ้นที่จอภาพ ผู้ใช้ที่เทอร์มินัลก็จะนับเงิน จ่ายอะไหล่ให้ลูกค้า เก็บเอกสารเข้าที่แล้วกดแป้นเพื่อยืนยันรายการ (Confirm) อีกครั้งหนึ่ง โปรแกรมได้รับสัญญาณยืนยันแล้วก็จะกลับไปสู่สภาพของเมนเมนูและรออยู่ที่ตรงจุดนั้นจนกว่าจะมีรายการใหม่เข้ามา

โปรแกรมที่จะทำงานให้ได้ตามลอจิกข้างบนนี้จะมีขนาดใหญ่ประมาณ 2 เมกะไบต์ ทั้งนี้โดยคิดว่าโปรแกรมนี้ทำได้ 10 ชนิดของรายการ และแต่ละชนิดกินเนื้อที่ประมาณ 200 เคไบต์ แต่เครื่องตามราคาที่พูดถึงจะมีเมมโมรี 8 เมกะไบต์ ใช้เป็นโอเปอเรติ้งซิสเท็มไป 2 เมกะไบต์ จึงเหลือให้ยูสเซอร์ได้ 6 เมกะไบต์ จากตัวเลขอันนี้เราก็บอกได้ว่ามีโปรแกรมเข้าไปอยู่ได้พร้อม ๆ กันไม่เกิน 3 เทอร์มินัล ส่วนที่เหลืออยู่อีก 7 เทอร์มินัลจะถูกเตะออกไปอยู่ในจานแม่เหล็ก ในสภาพแบบนี้สามารถบอกได้เลยว่าจะมีการสลับเข้าสลับออก (Swap-in Swap-out) ตลอดเวลา

จากนี้เราก็มาลองประมาณเวลาที่เครื่องจะโต้ตอบกับเทอร์มินัลในแต่ละขั้นตอนดู ในขณะที่โปรแกรมขึ้นภาพเป็นเมนเมนูอยู่นั้นซึ่งเรามักจะยังไม่ได้ทำอะไร โปรแกรมของเราจึงมักจะถูกโปรแกรมคนอื่นแย่งเมมโมรีไปใช้ ดังนั้นเมื่อกดแป้นเลือกชนิดของรายการเข้ามาเครื่องจะต้องรอเพื่อหาเมมโมรีให้ สมมติว่ามีโปรแกรมอื่นหยุดรอการอินพุตอยู่พอดี ดังนั้นโอเปอเรติ้งซิสเท็มจะทำการสลับที่เมมโมรีให้ เวลาส่วนใหญ่จะใช้ไปในการก็อบปี้โปรแกรมเก่าขึ้นจานแม่เหล็กและเอาโปรแกรมของเราจากจานแม่เหล็กลงมาในเมมโมรี โปรแกรมขนาด 2 เมกะไบต์จะใช้เวลาประมาณ 2 ถึง 10 วินาที ทั้งนี้ขึ้นอยู่กับว่าเรามีแคชเมมโมรี (Cache Memory) ช่วยหรือไม่ นอกจากนี้แล้วโอเปอเรติ้งซิสเท็มที่ฉลาดหน่อยก็จะไม่โหลดทั้งหมดมาให้ เขาแบ่งโปรแกรมของเราออกเป็น "หน้า" การโหลดกลับลงมานี้จะเอาแต่เฉพาะหน้าที่ต้องการเท่านั้น วิธีนั้นจะลดเวลาของการสลับที่บนเมมโมรีเหลือแค่ 0.2-1.0 วินาที หลังจากนั้นก็จะส่งภาพเมนูของรายการที่ต้องการออกมา ซึ่งเรื่องนี้จะช้าหรือเร็วจะขึ้นกับจำนวนตัวอักษรที่ส่งออกคนรุ่นใหม่มักจะส่งออกมาเต็
จอภาพ คือ 2000 ตัวอักษร ซึ่งจะ
เวลาประมาณ 2000×8/9600 = 1.6
วินาที นี่เพียงคิดที่สปีด 9600 บิตต่
1 วินาที และคิดว่าไม่ต้องมีอะไรปะหน้
ปะท้ายข้อความนั้นเลย สรุปแล้วการตอ
ครั้งแรกใช้เวลาประมาณ 2 วินาทีไ
จนถึง 12 วินาที ทั้งนี้ขึ้นอยู่กับความ
ชาญฉลาดของโอเปอเรติ้งซิสเท็ม

จะขอผ่านไปถึงจังหวะที่มีการ
ส่งข้อมูลเข้ามาเพื่ออัพเดตเลยดีกว่า
ขั้นตอนดังกล่าวจะต้องมีการส่งข้อมู
เข้ามาแล้วคำนวณและอัพเดตจานแม่เหล็
และส่งผลกลับออกไปที่จอภาพ สมมุ
ว่าข้อมูลส่งเข้ามายาว 500 ตัวอักษ
และการอัพเดตนั้นต้องอ่านและบันทึ
จานแม่เหล็ก 12 ครั้ง และส่งผลกลับ
ขึ้นจอภาพเป็นจำนวน 2000 ตัวอักษ
เวลาที่ส่งข้อมูลเข้ามา = 500×8/960
= 0.42 วินาที เวลาที่อ่านและบันทึกจาน
แม่เหล็ก = 12×20 มิลลิ = 0.24 วินา
เวลาที่ใช้ในการคำนวณ = 200,000
2,000,000 = 0.10 วินาที เวลาที่ส
ข้อมูลออก = 1.67 วินาที และเวลา
ต้องสลับเมมโมรี = 1.0 ถึง 10 วินา
ซึ่งขึ้นอยู่กับจำนวนหน้ารวมเวลาทั้งหม
= 3.43 ถึง 12.43 วินาที

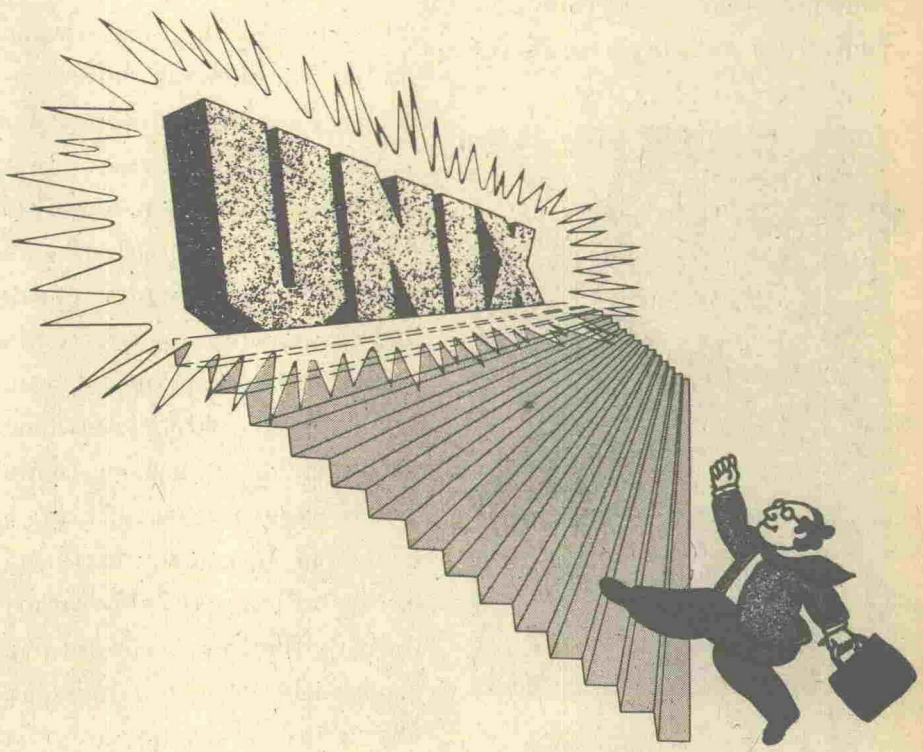
นั่นเป็นเวลาที่เครื่องจะโต้ตอ
(Response Time) ได้ในจังหวะที่เป็
งานหลัก คือจังหวะที่ต้องอัพเดตข้อมู
แต่จะเป็นประมาณการทางด้านต่ำ ทั้ง
เพราะการอ่านและบันทึกจานแม่เหล็
นั้นอาจต้องมีการรอคอยกัน (I/O Wa
และการคำนวณนั้นก็อาจต้องรอคอย
การใช้ซีพียู (CPU Wait) ทั้งนี้เพรา
เราไม่ใช่โปรแกรมเดียวที่เข้าไปวิ่งอยู่ใ
เครื่อง เวลาที่ต้องรอคอยทั้ง 2 ชนิด
จะขึ้นอยู่กับอัตราการใช้จานแม่เหล็

และซีพียู (Disk Utilization Factor และ CPU Utilization Factor) เรื่องนี้มีคนเขาศึกษาและทำเป็นกราฟและตารางเอาไว้ให้เราใช้ ตัวอย่างเช่น ของเจมส์-มาร์ติน เป็นต้น ใครอยากศึกษาอย่างละเอียดก็ไปค้นหามาอ่านกันได้ แต่เพื่อให้บทความนี้จบลงได้ง่าย ๆ จะบอกว่าเครื่องระดับ 80386 นี้จะทำให้การอ่านและบันทึกจานแม่เหล็กยืดออกไปอีกประมาณเท่าตัว และเวลาในการคำนวณก็ยืดออกไปประมาณ 2 เท่าตัว ดังนั้นเวลาที่เครื่องใช้จริง ๆ จะเป็นประมาณ 0.42+0.42×2+0.10×3+1.67+1.0 = 3.87 ถึง 12.87 วินาที

ปัญหาของเครื่องประเภทมัลติ-ยูสเซอร์นี้ยังต้องระวังอีก 2 อย่าง ประการแรกคือ ในตอนที่รับข้อมูลเข้ามาและอีดิติ้งนั้นถ้ามีการอ่านข้อมูลจากจานแม่เหล็กขึ้นมาใช้ช่วยในการอีดิติ้งก็จะทำให้เรคอร์ดเหล่านั้นถูกล็อกเอาไว้ เมื่อส่งผลออกไปที่จอภาพเพื่อให้ผู้ปฏิบัติงานต้องตอบยืนยัน (Confirm) ก็ต้องรอ การรอคอยนี้จะนาน 5 ถึง 10 วินาที ซึ่งในตอนนั้นโปรแกรมอื่นจะเรียกใช้เรคอร์ดเหล่านี้ไม่ได้เลย เทอร์มินัลอื่นอาจต้องรอคอยตามไปด้วย เวลารอคอยดังกล่าวนี้จะเกิดขึ้นเมื่อต้องใช้เรคอร์ดเดียวกัน ผู้ออกแบบระบบจะต้องพยายามหลีกเลี่ยงไม่ให้มีเรคอร์ดประเภทร่วม (Common Record) เช่น เรคอร์ดที่เก็บหมายสลิปที่จะใช้ต่อ ๆ กันไป ฯลฯ

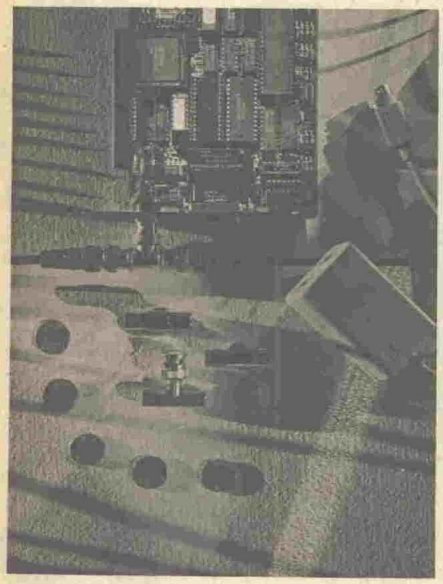
เรื่องที่ 2 ที่ผู้ออกแบบระบบต้องระวังก็คือการสลับที่เมมโมรี (Virtual Call หรือ Swap-In Swap-Out) นั้นจะใช้เวลามากขึ้นเมื่อจานแม่เหล็กเริ่มทำงานจนไม่ได้หยุด (Busy) เพราะเมื่อโหลดกลับเข้ามาได้แล้วก็อาจทำการคำนวณเพียงเล็กน้อยแล้วก็ต้องถูกเตะออกไปอีก ทั้งนี้เพราะมีการสั่งให้อ่านหรือบันทึกลงจานแม่เหล็ก การสลับกันไปสลับกันมานี้อาจสูงจนถึงขั้น "ไม่รู้จักหยุดหย่อน" ซึ่งจะทำให้ทั้งระบบทำงานช้าลงอย่างน่าใจหาย ปัญหานี้เขารู้กันดีในเครื่องเมนเฟรมและมินิคอมพิวเตอร์ แต่ผู้ผลิตซอฟต์แวร์ระดับมัลติยูสเซอร์มักจะมองข้าม เพราะคิดว่าระบบของตนจะถูกใช้ในเน็ตเวอร์กเล็ก ๆ แค่ 6-7 เทอร์มินัล หรือถ้าจะเป็นเน็ตเวอร์กขนาดใหญ่เขาก็คิดว่าจะเป็นผู้ใช้ประเภท "นักวิจัยและอาจารย์" ซึ่งทำรายการไม่มากเท่ากับงานดาต้าโปรเซสซิ่งทั่ว ๆ ไป ปัญหานี้จึงไม่มีใครคิดและป้องกันไว้ จุดที่จะเกิดปัญหาสะแวปปิ้ง (Swapping In/Out) นี้พอจะประมาณได้ดังนี้

สมมติว่าเราเพิ่มเทอร์มินัลเป็น n เทอร์มินัล และสมมติว่า 1 รายการจะมีการอ่านและบันทึก 12 ครั้ง (ตามลอจิกของโปรแกรมที่ดี) ในการอ่านและบันทึกแต่ละครั้งเราจะมีโอกาสถูกเตะออกไปจากเมมโมรี เรามาสมมติกันว่าถูกเตะออกไป 2 ครั้ง ดังนั้นจึงต้องมีการโหลดใหม่ประมาณ 400 เคไบต์ (รูทีนที่ทำหน้าที่อัพเดตตามรายการนี้เท่ากับ 200 เคไบต์) แต่ก่อนจะเอาส่วนของโปรแกรมนี้ลงมาก็ต้องก๊อบปี้ส่วนของโปรแกรมคนอื่นขึ้นไปเป็นจำนวนเท่า ๆ กัน ดังนั้นการทำสะแวปปิ้ง 2 ครั้งจะมีผลเท่ากับอ่านและบันทึก 800 เคไบต์ การอ่านและบันทึกนี้จะเป็นจำนวน = 800/4 = 200 ครั้ง ทั้งนี้โดยสมมติว่าดิสก์ของเราจัดเป็นบล็อกละ 4 เคไบต์ ดังนั้นการทำรายการ 1 รายการจะต้องอ่านและบันทึกจานแม่เหล็ก 200+12 = 212 ครั้ง แต่เรามี n เทอร์มินัล และแต่ละเทอร์มินัลจะทำรายการได้ประมาณ 1 รายการต่อ 1 นาที ดังนั้นใน 1 นาทีจะต้องอ่านและบันทึกจาน-แม่เหล็ก 212 n ครั้ง ตัวเลขนี้จะต้องนำไปเข้าสมการให้เท่ากับความเร็วของจานแม่เหล็ก จานแม่เหล็กจะอ่านหรือ



บันทึกได้ 50 ครั้งต่อ 1 วินาที หรือ 50×60 = 3000 ครั้ง ต่อ 1 นาที ดังนั้น n = 3000/2112 = 14.1

จากตัวเลขข้างบนนี้จะเห็นว่าถ้ามีการทำสะแวปปิ้ง 2 ครั้งต่อ 1 รายการ จะทำให้ใส่เทอร์มินัลได้ไม่เกิน 14.1 ตัว ปัญหานี้เขารู้กันในหมู่ผู้ใช้เครื่องเมนเฟรมและมินิคอมพิวเตอร์ วิธีแก้ปัญหานั้นมีหลายวิธี เช่น ต้องใช้คอมพายเลอร์ที่คอมพายล์ออกมาเป็นโปรแกรมชนิดเริ่มต้นใหม่ได้ทุกที่ (Re-entrance Code) และจะทำสะแวปปิ้งแต่เฉพาะส่วนที่เป็นคำสั่ง (Executable Code) เขาจะไม่สะแวปปิ้งพวกที่เป็นข้อมูล (Data Area) วิธีดังกล่าวนี้ทำให้ไม่ต้องก๊อบปี้โปรแกรมขึ้นจานแม่เหล็ก ซึ่งจะลดจำนวนการอ่านและบันทึกจาก 200 ครั้งเหลือ 100 ครั้ง จำนวนเทอร์มินัลก็จะเพิ่มขึ้นได้อีกเกือบเท่าตัว วิธีแก้ไขปัญหาแบบที่ 2 คือลดขนาดของโปรแกรมลงไป เช่น ถ้าลดลงได้ครึ่งหนึ่งก็จะเพิ่มเทอร์มินัลได้อีกเกือบเท่าตัว แต่วิธีนี้ใช้ได้กับคนรุ่นเก่าเท่านั้น คนรุ่นใหม่เขาชอบพูดกัน



เป็นเม็ก ๆ ใครเขียนโปรแกรมเล็กแสดงว่าไม่เก่ง ในระบบที่ยกเป็นตัวอย่างนี้เราจะเห็นว่าถ้าลดขนาดของโปรแกรมลงเหลือ 1 เมกะไบต์ และเพิ่มเมมโมรีของเครื่องเป็น 16 เมกะไบต์ก็จะวิ่งได้ 14 เทอร์มินัลโดยไม่เกิดการสะแวปปิ้งเลย ซึ่งจะได้ความเร็วในการโต้ตอบครั้งละ 2-3 วินาทีเท่านั้น

จากตัวเลขที่กล่าวมาข้างบนนี้เราจะเห็นว่า การย้ายงานจากเครื่องเมนเฟรมและมินิคอมพิวเตอร์ไปลงระบบมัลติ-ยูสเซอร์นั้นสู้ระบบแลนไม่ได้ มันสู้กันไม่ได้ทั้งในเรื่องค่าใช้จ่ายและความเร็วในการโต้ตอบ (Response Time) แต่ใคร่ขอย้ำอีกครั้งหนึ่งว่า นี่เป็นงานประเภทธุรกิจทั่วไปและมีการอัพเดตที่บ่อยมาก เช่นวันละ 5,000-50,000 รายการ งานพวกนี้แตกต่างจากงานค้นคว้าและวิจัยของอาจารย์และนักวิจัย

## สรุป

โดยสรุปแล้วแลนนับเป็นทางเลือกที่ดีที่สุดในการย้ายงานจากเครื่องเมน-เฟรมและมินิคอมพิวเตอร์ แต่ท่านจะต้องระวังอยู่ 2-3 เรื่อง เรื่องแรกคืออย่าออกแบบระบบให้โปรแกรมไปล็อกไฟล์หรือเรคอร์ดเอาไว้นาน เรื่องที่ 2 คืออย่าออกแบบระบบที่ต้องส่งข้อมูลผ่านเคเบิลของแลนเกิน 200,000 ไบต์ต่อ 1 วินาที และเรื่องที่ 3 คืออย่าออกแบบระบบที่ต้องอ่านและบันทึกลงจานแม่เหล็กที่ตัวเซอร์ฟเวอร์เกิน 200,000 ไบต์ต่อ 1 วินาที จุดคอขวด 2 เรื่องหลังนี้จริง ๆ แล้วอยู่สูงกว่าที่เราต้องการใช้งาน แต่ก็มีภาษาดาต้าเบสบางชนิดใช้ทรัพยากรเหล่านี้ฟุ่มเฟือยมาก ดังนั้นก่อนที่จะเอาแลนไปใช้งานควรศึกษาให้ละเอียด ข้อจำกัดอันที่ 3 นั้นอาจพูดอีกแบบหนึ่งได้ คือควรออกแบบระบบให้อ่านและบันทึกจานแม่เหล็กประมาณ 12 ครั้งต่อ 1 รายการ และถ้าจำเป็นจริง ๆ ก็อาจเพิ่มได้ถึง 200 ครั้งต่อ 1 รายการ ณ จุดสูงสุดนี้จะทำให้เราทำงานได้เพียง 6-7 เทอร์มินัลพร้อม ๆ กัน

ก่อนจะจบบทความนี้ ใคร่ขอเรียนให้ทราบว่าในปัจจุบันนี้เขาพัฒนาให้แลนใช้เส้นใยแสงเป็นตัวเคเบิลสื่อสารและสามารถเพิ่มจำนวนไดรฟ์และจำนวนตัวควบคุมจานแม่เหล็กได้ ดังนั้นระบบแลนจึงไม่มีขีดจำกัดเหลืออยู่เลย ใครที่รู้วิธีออกแบบระบบไม่ให้มีการล็อกไฟล์และล็อกเรคอร์ดเอาไว้นาน ๆ ก็จะใช้กับงานในระดับเป็นพัน ๆ เทอร์มินัลได้ เทอร์มินัลเหล่านี้สามารถทำงานในระดับความฉี่ของภาพวิดีโอได้ คือสามารถรับส่งภาพกราฟิกและวิดีโอไป-มาได้โดยสะดวก เทคโนโลยีใหม่อันนี้เขาเรียกว่า มัลติมีเดีย (Multi-Media) ซึ่งจะเป็นเทคโนโลยีแห่งทศวรรษนี้